Core Python Programming

Sept. 7, 2016

1. **Python Language Basics**
    a. using conda to manage Python environments & packages
    b. standard Python keywords, rules for identifiers, & naming conventions
    c. Python idioms for importing modules, data, and functions into a Python program
    d. using help (and other documentation) to learn about builtin functions
    e. standard constructs for flow control: for, while, if-elif-else, etc.
    f. functions to reproduce complicated sequences of computations
    g. define functions with keyword and default arguments
    h. building lists, sets, dictionaries and tuples
    i. Python idioms for iteration data structures
2. **Python Language Basics: Deeper Knowledge**
    a. apply common methods associated with builtin Python data types
    b. apply the str.format mini-language to generate formatted output
    c. apply Python rules for indexing & slicing strings, lists, & tuples
    d. use comprehensions to replace complex nests of loops & conditionals
3. **The Python Standard Library**
    a. Search the documentation of the Python Standard Library for useful modules
    b. import the datetime module and perform operations with datetime objects.
    c. import the collections module and utilize Counter and defaultdict container objects
4. **Numpy for numeric data**
    a. Numpy basics
        i. create NumPy arrays of zeros, ones, ranges and random numbers
        ii. *dtypes*
        iii. the NumPy shape attribute and how to reshape an array
    b. NumPy arrays
        i. indexing N-dimensional arrays by position
        ii. extracting data along each array dimension
        iii. boolean and complex selections
        iv. differences between *views* and *copies*
    c. NumPy computation
        i. vectorized computation on an N-dimensional array
        ii. aggregations (sum, mean, std, etc.) along each axis
        iii. vectorized functions that take NumPy arrays as input
        iv. *broadcasting* rules for computation between arrays with differing dimensionalities

Data Centric Computing with Python on Peregrine

Sept. 13, 2016

1. **Review conda environments**
2. **Run Jupyter Notebooks from an interactive batch job**
    a. https://github.com/AlbertDeFusco/qJuypter
3. **Pandas + Xarray: tabular and high-dimensional data**
4. **Plotting**
    a. Matplotlib review
    b. Interactive plotting (Bokeh, Plot.ly)
    c. Dashboards and data streaming

Performance Computing with Python on Peregrine

Sept. 14, 2016

1. **Python performance optimization overview**
2. **Utilizing profilers and debuggers**
3. **Get the most out of Numpy**
    a. MKL threads with Anaconda
    b. Vectorization
    c. The numexpr module
4. **Compiled optimizations**
    a. Numba
    b. Cython
    c. Other compiled-code interfaces
5. **Parallelization in Python**
    a. Dask array and dataframe
        i. Automatic chunking for parallel computation
        ii. Out-of-core operations for large datasets
    b. Embarrassingly parallel
        i. multiprocessing module
        ii. concurrent.futures ProcessPool
    c. Multi-node parallelization
        i. Dask + Distributed in batch systems; including Numba
        ii. MPI4Py
            1. Including single-node parallelization with Dask/Numba/Cython
            2. Parallel I/O with HDF5